RDF as a Data Structure for Software Engineering

Axel Rauschmayer, axel@rauschma.de
Institut für Informatik, Ludwig-Maximilians-Universität München

Abstract and Introduction

Software engineers encounter numerous models in their day-to-day work: the source code, documentation, requirements, bug tracking, a web site used for public collaboration etc. Currently, many ad-hoc ways exist to manage all these models in an integrated fashion. Hyena is a platform for *integrated software engineering model management* (ISEMM) that uses RDF for principled model manipulation and storage. It brings out a different side of RDF, where one is less concerned with its knowledge representation abilities and more focused on using it as a graph-based data structure.

Requirements for an ISEMM Platform

For each model one encounters in ISEMM, model-specific support must be implemented first at the API level to make programming with the model easier. Second, one provides the end user with a graphical interface for editing the model. Once support for the separate models is complete, there need to be ways to integrate them, especially where they influence each other. Finally, as software engineering is often about communication, it should be easy to publish and collaboratively edit models. All of these tasks need to be facilitated by an ISEMM engine. When it comes to generic model editing the following operations are frequently used and are thus requirements for the ISEMM data structure.

- Modularization: is one crucial element for managing complexity in software engineering. Accordingly, programming languages feature many modularization techniques, but few of them extend to artifacts other than code. Modularization should permit us to join and separate modules/models at will which lets us explore concerns (encapsulated in modules) both separately and combined.
- Annotation: For documentation, categorization etc., one often has to add new information to models *anywhere* and without interfering with their processing. Note that this can also be considered a modularization problem, the annotated data being a new model that is layered on top of others.
- Querying: Having integrated several models, we would like to examine the integrated information space: filtering, projections and browsing should allow us to discover concerns that cut across models which further extends our concern exploration abilities.
- Multi-dimensional categorization: with *multi-dimensional separation of concerns*, software engineering recognized the fact that a software system is a multi-dimensional space. All functionality related to a concern cuts across it as a hyper-plane. Having the ability to make these hyper-planes explicit (to *reify* them) is important for organizing the system. Thus, any categorization has to be multi-dimensional where artifact kind is just another dimension.

A Platform for Software Engineering

The cornerstone of HYENA is its universal data structure, RDF. It supports modularization via dynamic merging of models. Using a shared vocabulary for nodes and separate vocabularies for predicates, one can integrate models while their processing stays independent. Everything can be annotated in RDF, because any structure can be reified. RDF query languages such as SPARQL provide formally sound query mechanisms. Multi-dimensional categorization is possible because RDF's relational nature is inherently multi-dimensional. Finally, URI-based nodes make it easy to integrate many kinds of artifacts. The remaining requirements are met by the following parts of HYENA:

- *Vodules* (vocabulary modules): provide complete support for one kind of model. In addition to declarative data from an RDF ontology, vodules support model editing via an API, GUI components and/or web services.
- Application programming interface (API): The *core API* of HYENA provides infrastructure functionality, everything else is added via vodules. Vodules are thus both clients and providers of API functionality. The API of HYENA is even useful for stand-alone programs.
- Web service framework: Each model managed by HYENA can be exposed as a web service, in a model-dependent manner. For example, an RDF model encoding a wiki web site will be displayed as a series of web pages.
- *Graphical user interface* (GUI) components: HYENA is itself integrated in the *Eclipse integrated development environment* and extends rather than replaces its facilities. RDF-specific functionality such as bookmarks and interactive filtering are the foundation on which vodules base their own GUI contributions. These contributions are essential for shielding the user from many complexities of RDF when editing a custom model.

Vodules that Come Packaged with Hyena

HYENA's standard vodules make it a useful software engineering environment right out of the box:

- JTUBE, Java source code management: makes Java artifacts accessible to HYENA. Other models can now provide a lightweight semantic layer on top of Java source code. JTUBE tracks changes to the code and co-evolves linked models. Furthermore, JTUBE allows one to move back and forth between code and associated RDF data in Eclipse.
- PEERSTORM, collaborative conceptual brainstorming: enables a group of people to work together on conceptual sketches such as requirements, design ideas, outlines and task lists. Editing is distributed and changes are displayed in real time. Concepts are expressed as *snippets*, an RDF-encoded data structure that is similar to topic maps.
- WIKKED, web-based publishing and editing: a vodule for storing a wiki in an RDF graph. Displaying and editing the wiki is implemented as a web service, as is uploading of new data. JTUBE uses WIKKED to publish Java source code; PEERSTORM displays snippets in various ways.